Package RMiner : Data Mining Classification et Méthode de Régression

Rminer : kesako ?

Le package Rminer aide à la manipulation des données algorithmiques pour le data mining concernant la classification et la régression (incluant la prévision de series temporelles).
Pour utiliser le package Rminer et ses fonctions :

- vérifier si le package est installé dans R

- sinon installer le package puis l'appeler avec la library

`Install.packages(« rminer »)`

`Library(rminer)`

Les fonctions de Rminer

Ci-desous la liste des fonctions du package Rminer. Nous n'allons vous présenter que les fonctions qui nous semblent les plus pertinentes dans ce tutoriat.
Si vous souhaitez approfondir votre connaissance sur les autres fonctions, nous vous invitons à aller sur la documentation R du package [Lien]

Si vous souhaitez consulter l'aide, taper dans la commande R `help(package=rminer)`

Liste des fonctions de Rminer :

CasesSeries
crossvaldata
delevels
fit
holdout
Importance
imputation
lforecast
mgraph
mining
mmetric
mparheuristic
predict.fit
savemining
sa_fri1
sin1reg
vecplot

Ce tutoriat se concentra sur les fonctions suivantes du package Rminer qui peuvent être réparties dans 3 phases dans l'analyse des données :
- Data preparation : CasesSeries, delevels
- Modeling : fit, predict, mining
- Evaluation : mmetric, mgraph, mining

1. Data preparation : CasesSeries, delevels

`*CasesSeries*` : créer un data frame d'une série temporelle en utilisant la fenêtre glissante. Fenêtre glissante : (« sliding window ») : Fenêtre temporelle utilisée dans l'exploration des flots de données (« Data stream mining ») pour en extraire des motifs. Les fenêtres peuvent avoir une taille W fixe et la fouille de données s'effectue sur les W dernières transactions, ou sur les W dernières unités de temps, elles peuvent aussi avoir une taille variable, dans la détection de la dérive conceptuelle.

 # Exemple

```r
t=1:20
d=CasesSeries(1:10,c(1,3,4))
print(d)
d=CasesSeries(1:10,c(1,2,3))
print(d)
```

```
lag4 lag3 lag1  y
1   1   2   4  5
2   2   3   5  6
3   3   4   6  7
4   4   5   7  8
5   5   6   8  9
6   6   7   9 10

  lag3 lag2 lag1  y
1   1   2   3  4
2   2   3   4  5
3   3   4   5  6
4   4   5   6  7
5   5   6   7  8
6   6   7   8  9
7   7   8   9 10
```

==> la fonction retourne un data frame où y est cible de l'output et les inputs sont le temps de latence

`*Delevels`:* réduire, remplace ou transforme les niveaux du data frame et la variable du facteur.

```r
f=factor(c("A","A","B","B","C","D","E"))
print(table(f))
```

```
f
A B C D E
2 2 1 1 1

# remplacer "A" en "a":
f1=delevels(f,"A","a")
print(table(f1))
f1
a B C D E
2 2 1 1 1
```

```{r} # combiner c("C","D","E") en "CDE"
f2=delevels(f,c("C","D","E"),"CDE")
print(table(f2))
f2
 A   B CDE
 2   2   3

# combiner c("B","C","D","E") en _OTHER:
f3=delevels(f,c("B","C","D","E"))
print(table(f3))
f3
    A _OTHER
    2     5
```

2. Modeling : fit, predict

Rminer comprend 14 méthodes de classification et 15 méthodes de régression, tout directement disponible à travers les fonctions `fit, predict, mining`.

`*Fit*` : la fonction ajuste un modèle sélectionné de jeux de données et peut automatiquement ajuster les hyperparamètres. Les hyperparamètres sont des paramètres réglables qui vous permettent de contrôler le processus d'entraînement du modèle.

`*Predict*` : la fonction donne un modèle ajusté et calcule les prédictions pour un jeu de données.

Les fonctions `*fit*` et `*predict*` proposent plusieurs modèles tels que :
- rpart (arbre de décision)
- randomForest
- lm (régression linéaire ou multiple)
- cv.glmnet : modèle linéaire généralisé

Par défaut, le type du modèle Rminer (classification ou régression) dépend du type d'output.
- Si c'est un facteur (discret) alors c'est une probabilité de classification
- Si c'est numérique (int, num) alors c'est une régression qui est exécutée.

Voir tous les modèles de `fit` et `predict` : [lien fit]

 # Exemple qui montre comment la transformation fonctionne avec `fit` et `predict`

```{r}
M=fit(y~.,data=sa_ssin,model="mr") # linear regression
P=predict(M,data.frame(x1=-1000,x2=0,x3=0,x4=0,y=NA)) # P should be negative
print(P)
[1] -0.4144042

M=fit(y~.,data=sa_ssin,model="mr",transform="positive")
P=predict(M,data.frame(x1=-1000,x2=0,x3=0,x4=0,y=NA)) # P is not negative
print(P)
[1] 0
```

3. Evaluation : mmetric, mgraph, mining

Rminer comprend une large sélection de métriques d'évaluation et de graphes qui peut être utilisée pour évaluer la qualité des modèles ajsutés et extraire les données apprises du modèle data-driven.

`*Mmetric*` : ffonction qui calcule les erreurs métriques de classification ou régression.

Ci-dessous les quelques mesures de `*mmetric*` :
- ALL : sort toutes les mesures de `*mmetric*`
- F1 score, [0-100%]
- TPR (true positive rate) : c'est la sensibilité, correspondant au taux de vrais positifs, 0-100%]
- PRECISION : la précision, c'est-à-dire le nombre de documents pertinents retrouvés rapporté au nombre de documents total proposé pour une requête donnée.
- ACC : taux d'exactitude de classification, [0-100%]
- ACCLASS : taux d'exactitude de classification par classe, [0-100%]
- TPR (true positive rate) : c'est la sensibilité, correspondant au taux de vrais positifs, 0-100%]

Voir tous les modèles de `*mmetric*` : [Lien]

 # Exemple

```{r}
y=factor(c("a","a","a","a","b","b","b","b"))
x=factor(c("a","a","b","a","b","a","b","a"))
print(mmetric(y,x,"CONF")$conf)
     pred
target a b
    a 3 1
    b 2 2

print(mmetric(y,x,metric=c("ACC","TPR","ACCLASS")))
    ACC    TPR1    TPR2 ACCLASS1 ACCLASS2
   62.5    75.0    50.0    62.5    62.5
```

```
print(mmetric(y,x,"ALL"))
      ACC        CE       BER     KAPPA    CRAMERV
62.5000000 37.5000000 37.5000000 25.0000000  0.0000000
  ACCLASS1   ACCLASS2   BAL_ACC1   BAL_ACC2       TPR1
62.5000000 62.5000000 62.5000000 62.5000000 75.0000000
     TPR2       TNR1       TNR2 PRECISION1 PRECISION2
50.0000000 50.0000000 75.0000000 60.0000000 66.6666667
      F11        F12       MCC1       MCC2
66.6666667 57.1428571  0.5163978  0.5163978
```

`*Sin1reg*` : un simple jeu de données avec 1000 points où y = 0.7 * sin(pi * x1/2000) + 0.3 * sin(pi * x2/2000)

`*Mgraph*` : fonction graphique

`*Mining*`: la fonction réalise plusieurs exécutions ajustées et de prédictions, selon une méthode de validation et donne un nombre d'exécutions.

 # Exemple de régression avec les fonctions sin1reg, mgraph et mining

```{r}
data(sin1reg)
M1=mining(y~.,sin1reg[,c(1,2,4)],model="mr",Runs=5)
M2=mining(y~.,sin1reg[,c(1,2,4)],model="mlpe",nr=3,maxit=50,size=4,Runs=5,feature="simp")
L=vector("list",2); L[[1]]=M2; L[[2]]=M1
mgraph(L,graph="REC",xval=0.1,leg=c("mlpe","mr"),main="REC curve")
mgraph(L,graph="DLC",metric="TOLERANCE",xval=0.01,
     leg=c("mlpe","mr"),main="DLC: TOLERANCE plot")
mgraph(M2,graph="IMP",xval=0.01,leg=c("x1","x2"),
     main="sin1reg Input importance",axis=1)
mgraph(M2,graph="VEC",xval=1,main="sin1reg 1-D VEC curve for x1")
mgraph(M2,graph="VEC",xval=1,
     main="sin1reg Courbe et histogramme pour x1",data=sin1reg)
```